# Assessing the Effectiveness of Acceleration Methods for Deterministic Neutron Transport Solvers

## Building a new tool for developers.

J. S. Rehak

ANS Summer Meeting: Acceleration Methods

June 10th, 2020

A majority of our limited time and effort (and funding) should be dedicated to designing new and better acceleration methods, **not implementing and analyzing results**.

# Outline

1. Why acceleration methods?
2. Analysis and implementation challenges
3. Design paradigm
4. Status and future work

# Steady-state Boltzman Transport Equation

Our problem of interest is the time-independent transport equation on a domain of interest $\mathbf{r} \in V$ [3],

$$\left[\hat{\Omega} \cdot \nabla + \Sigma_t(\mathbf{r}, E)\right] \psi(\mathbf{r}, E, \hat{\Omega}) \\ = \int_0^\infty dE' \int_{4\pi} d\hat{\Omega}' \Sigma_s(\mathbf{r}, E' \to E, \hat{\Omega}' \to \hat{\Omega}) \psi(\mathbf{r}, E', \hat{\Omega}') \\ + Q(\mathbf{r}, E, \hat{\Omega}) \ ,$$

with a given boundary condition,

$$\psi(\mathbf{r}, E, \hat{\Omega}) = \Gamma(\mathbf{r}, E, \hat{\Omega}), \quad \mathbf{r} \in \partial V, \quad \hat{\Omega} \cdot \hat{n} < 0$$

# Deterministic methods

Discretizations:

- Split up the spatial domain into cells.

# Deterministic methods

Discretizations:

- Split up the spatial domain into cells.
- Split up the energy domain into groups (multi-group equations).

# Deterministic methods

Discretizations:

- Split up the spatial domain into cells.
- Split up the energy domain into groups (multi-group equations).
- Solve along particular angles (collocation).

# Deterministic methods

Discretizations:

- Split up the spatial domain into cells.
- Split up the energy domain into groups (multi-group equations).
- Solve along particular angles (collocation).

$$\mathbf{L}\mathbf{\Psi} = \mathbf{M}\left[\mathbf{S} + \frac{1}{k}\mathbf{F}\right]\mathbf{\Phi}$$

# Deterministic methods

Discretizations:

- Split up the spatial domain into cells.
- Split up the energy domain into groups (multi-group equations).
- Solve along particular angles (collocation).

$$\mathbf{L}\mathbf{\Psi} = \mathbf{M}\left[\mathbf{S} + \frac{1}{k}\mathbf{F}\right]\mathbf{\Phi}$$

Gauss-Seidel source iteration $\mathbf{\Psi}_{(k+1)} = \mathbf{L}^{-1}\mathbf{M}\left[\mathbf{S}\mathbf{\Phi}_{(k)} + \frac{1}{k}\mathbf{F}\mathbf{\Phi}_{(0)}\right]$

Power iteration $\mathbf{\Psi}_{(k+1)} = \mathbf{L}^{-1}\mathbf{M}\left[\mathbf{S}\mathbf{\Phi}_{(0)} + \frac{1}{k}\mathbf{F}\mathbf{\Phi}_{(k)}\right]$

# Convergence challenges

## Convergence of Source Iteration

Gauss-Seidel source iteration can converge arbitrarily slowly as $\Sigma_s/\Sigma_t$ approaches unity.

# Convergence challenges

## Convergence of Source Iteration

Gauss-Seidel source iteration can converge arbitrarily slowly as $\Sigma_s/\Sigma_t$ approaches unity.

## Convergence of Power Iteration

Power iteration can converge arbitrarily slowly as the dominance ratio $k_1/k_0$ approaches unity.

# Convergence challenges

## Convergence of Source Iteration

Gauss-Seidel source iteration can converge arbitrarily slowly as $\Sigma_s/\Sigma_t$ approaches unity.

## Convergence of Power Iteration

Power iteration can converge arbitrarily slowly as the dominance ratio $k_1/k_0$ approaches unity.

Motivates the development of **acceleration methods** to address these issues.

- Source Iteration: Diffusion two-grid method (TG).
- Power Iteration: Nonlinear diffusion acceleration (NDA).

# Defining acceleration

# Defining acceleration

## Primary goal

To reduce the total amount of computational work required for an iterative method to converge[1].

# Defining acceleration

### Primary goal

To reduce the total amount of computational work required for an iterative method to converge[1].

The error in step $\ell$, is given by

$$\mathbf{e}_{(\ell)} = \mathbf{\Psi}^* - \mathbf{\Psi}_{(\ell)} \ .$$

# Defining acceleration

## Primary goal

To reduce the total amount of computational work required for an iterative method to converge[1].

The error in step $\ell$, is given by

$$\mathbf{e}_{(\ell)} = \mathbf{\Psi}^* - \mathbf{\Psi}_{(\ell)} \ .$$

Our method converges in $N$ steps when

$$\left\| \mathbf{e}_{(N)} \right\| < \varepsilon \ ,$$

# Defining acceleration

## Primary goal

To reduce the total amount of computational work required for an iterative method to converge[1].

The error in step $\ell$, is given by

$$\mathbf{e}_{(\ell)} = \mathbf{\Psi}^* - \mathbf{\Psi}_{(\ell)} \ .$$

Our method converges in $N$ steps when

$$\left\| \mathbf{e}_{(N)} \right\| < \varepsilon \ ,$$

with total computational work

$$W = \sum_{\ell=1}^{N} w_{(\ell)} \ .$$

# Defining acceleration

An accelerated method seeks to reduce the total computational work to achieve the same convergence. It is effective if

$$\sum_{\ell=1}^{N'} w'_{(\ell)} < \sum_{\ell=1}^{N} w_{(\ell)}$$

# Defining acceleration

An accelerated method seeks to reduce the total computational work to achieve the same convergence. It is effective if

$$\sum_{\ell=1}^{N'} w'_{(\ell)} < \sum_{\ell=1}^{N} w_{(\ell)}$$

**Note**

The same amount of error needs to be removed for the problem to converge, regardless of the method used to remove it.

# Outline

# Implementation

# Implementation

## Challenge

We need to modify an existing code, or create a new code to test our acceleration method.

# Implementation

## Challenge

We need to modify an existing code, or create a new code to test our acceleration method.

- Production codes can be difficult to modify.

# Implementation

## Challenge

We need to modify an existing code, or create a new code to test our acceleration method.

- Production codes can be difficult to modify.
- Writing new codes can be time consuming and costly.

# Implementation

## Challenge

We need to modify an existing code, or create a new code to test our acceleration method.

- Production codes can be difficult to modify.
- Writing new codes can be time consuming and costly.
- Reproducibility is difficult.

# Implementation

### Challenge

We need to modify an existing code, or create a new code to test our acceleration method.

- Production codes can be difficult to modify.
- Writing new codes can be time consuming and costly.
- Reproducibility is difficult.

# Implementation

## Challenge

We need to modify an existing code, or create a new code to test our acceleration method.

- Production codes can be difficult to modify.
- Writing new codes can be time consuming and costly.
- Reproducibility is difficult.

## What do we need?

A coding framework designed with the developer end-user in mind, that is portable and reproducible.

# Defining work

# Defining work

In general, we use inversions of the transport matrix – explicitly or implicitly (*sweeps*) – as a unit of work.

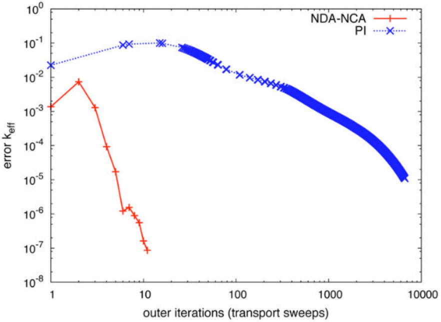



**Figure 1:** NDA convergence vs standard power iteration [7]

# Analysis challenges

# Analysis challenges

**Challenge**

Work definition requires (good) assumptions about algorithm efficiency.

# Analysis challenges

## Challenge

Work definition requires (good) assumptions about algorithm efficiency.

$$\sum_{\ell=1}^{N'} w'_{(\ell)} < \sum_{\ell=1}^{N} w_{(\ell)}$$

# Analysis challenges

## Challenge

Work definition requires (good) assumptions about algorithm efficiency.

$$\sum_{\ell=1}^{N'} w'_{(\ell)} < \sum_{\ell=1}^{N} w_{(\ell)}$$

$$N' w'_{(\mathsf{inv})} < N w_{(\mathsf{inv})}$$

# Analysis challenges

## Challenge

Work definition requires (good) assumptions about algorithm efficiency.

$$\sum_{\ell=1}^{N'} w'_{(\ell)} < \sum_{\ell=1}^{N} w_{(\ell)}$$

$$N' w'_{(\text{inv})} < N w_{(\text{inv})}$$

Relying on the number of sweeps as a measure of effectiveness relies on $w'_{(\text{inv})} \approx w_{(\text{inv})}$.

# Analysis challenges

**Challenge**

Work definition requires (good) assumptions about algorithm efficiency.

$$\sum_{\ell=1}^{N'} w'_{(\ell)} < \sum_{\ell=1}^{N} w_{(\ell)}$$

$$N' w'_{(\mathsf{inv})} < N w_{(\mathsf{inv})}$$

Relying on the number of sweeps as a measure of effectiveness relies on $w'_{(\mathsf{inv})} \approx w_{(\mathsf{inv})}$.

This becomes complicated as our acceleration methods become more complex, and take on more work.

# Validating our methods

# Validating our methods

**Challenge**

We want to validate *why* our methods work.

# Validating our methods

## Challenge

We want to validate *why* our methods work.

- The methods we develop are backed up by math and our understanding of the problem.

# Validating our methods

**Challenge**

We want to validate *why* our methods work.

- The methods we develop are backed up by math and our understanding of the problem.
- If a method is successful in accelerating a solve, it's not always clear why.

# Validating our methods

## Challenge

We want to validate *why* our methods work.

- The methods we develop are backed up by math and our understanding of the problem.
- If a method is successful in accelerating a solve, it's not always clear why.
- Combinations of methods may make the mathematical analysis difficult or impossible.

# Validating our methods

## Challenge

We want to validate *why* our methods work.

- The methods we develop are backed up by math and our understanding of the problem.
- If a method is successful in accelerating a solve, it's not always clear why.
- Combinations of methods may make the mathematical analysis difficult or impossible.

# Validating our methods

## Challenge

We want to validate *why* our methods work.

- The methods we develop are backed up by math and our understanding of the problem.
- If a method is successful in accelerating a solve, it's not always clear why.
- Combinations of methods may make the mathematical analysis difficult or impossible.

## What do we need?

Additional, good data that enables us to assess the effectiveness of our method.

# Analysis Challenges

A few challenges when analyzing the effectiveness of acceleration schemes include:

- Work definition requires assumptions about algorithm efficiency.

# Analysis Challenges

A few challenges when analyzing the effectiveness of acceleration schemes include:

- Work definition requires assumptions about algorithm efficiency.
- We need good data to show us *why* our methods are working.

# Analysis Challenges

A few challenges when analyzing the effectiveness of acceleration schemes include:

- Work definition requires assumptions about algorithm efficiency.
- We need good data to show us *why* our methods are working.
- Combined or complex schemes may invalidate assumptions.

# Analysis Challenges

A few challenges when analyzing the effectiveness of acceleration schemes include:

- Work definition requires assumptions about algorithm efficiency.
- We need good data to show us *why* our methods are working.
- Combined or complex schemes may invalidate assumptions.
- Implementation and reproducibility can be difficult.

# Outline

# Design goals for BART

The Bay Area Radiation Transport (BART) is motivated by three major design goals. To create a code that,

# Design goals for BART

The Bay Area Radiation Transport (BART) is motivated by three major design goals. To create a code that,

1. relieves some of the burden of implementing a novel acceleration method,

# Design goals for BART

The Bay Area Radiation Transport (BART) is motivated by three major design goals. To create a code that,

1. relieves some of the burden of implementing a novel acceleration method,
2. provides a controlled environment for measuring the effectiveness of the novel method, and,

# Design goals for BART

The Bay Area Radiation Transport (BART) is motivated by three major design goals. To create a code that,

1. relieves some of the burden of implementing a novel acceleration method,
2. provides a controlled environment for measuring the effectiveness of the novel method, and,
3. provides tools for verifying the basis for effectiveness.

# Designed for implementation

**Goal 1**

Relieving some of the burden of implementing a novel acceleration method.

# Designed for implementation

**Goal 1**

Relieving some of the burden of implementing a novel acceleration method.

BART is designed to be a code focused on a **developer** end-user.

# Designed for implementation

**Goal 1**

Relieving some of the burden of implementing a novel acceleration method.

BART is designed to be a code focused on a **developer** end-user.

- Focus on clarity of structure instead of performance (optimization for modification).

# Designed for implementation

**Goal 1**

Relieving some of the burden of implementing a novel acceleration method.

BART is designed to be a code focused on a **developer** end-user.

- Focus on clarity of structure instead of performance (optimization for modification).
- Heavy usage of polymorphism.

```
SystemInitializerI
------------------
+ Initialize(System&)
: void
```

# Designed for implementation

**Goal 1**

Relieving some of the burden of implementing a novel acceleration method.

BART is designed to be a code focused on a **developer** end-user.

- Focus on clarity of structure instead of performance (optimization for modification).
- Heavy usage of polymorphism.
- Comprehensive testing coverage.

# Controlled testing environment

**Goal 2**

Providing a controlled environment for measuring the effectiveness of the novel method.

# Controlled testing environment

**Goal 2**

Providing a controlled environment for measuring the effectiveness of the novel method.

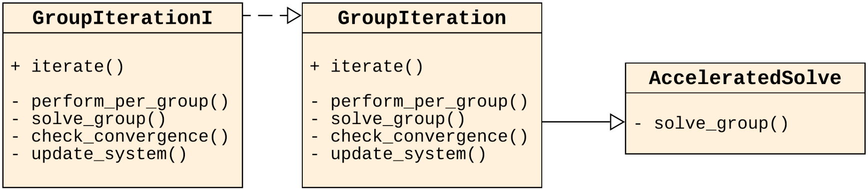
BART is designed to leverage polymorphism to isolate and minimize changes required to implement novel methods.

# Controlled testing environment

**Goal 2**

Providing a controlled environment for measuring the effectiveness of the novel method.

BART is designed to leverage polymorphism to isolate and minimize changes required to implement novel methods.



# Polymorphism Benefits

The use of polymorphism in BART

- Minimizes code changes needed to implement new methods, making it faster and easier.

# Polymorphism Benefits

The use of polymorphism in BART

- Minimizes code changes needed to implement new methods, making it faster and easier.
- Enables a true comparison of the accelerated solve to a control solve.

# Polymorphism Benefits

The use of polymorphism in BART

- Minimizes code changes needed to implement new methods, making it faster and easier.
- Enables a true comparison of the accelerated solve to a control solve.
- Makes the modifications *portable*.

# Polymorphism Benefits

The use of polymorphism in BART

- Minimizes code changes needed to implement new methods, making it faster and easier.
- Enables a true comparison of the accelerated solve to a control solve.
- Makes the modifications *portable*.
- Enables us to compare the implementation of the method to dis-aggregate the computer science from the method itself.

# Instrumentation

# Instrumentation

**Goal 3**

Provide tools for verifying the basis for effectiveness.

# Instrumentation

**Goal 3**

Provide tools for verifying the basis for effectiveness.

BART will include the ability to *instrument* a solve to gather enough data to draw useful conclusions about the effectiveness of acceleration schemes.

- Storage of solve parameters (eigenvalues, fluxes).
- Storage of hierarchy of iterations.
- Calculation and storage of error or residual.
- Analysis of Fourier error modes coefficients.

# Instrumentation

**Goal 3**

Provide tools for verifying the basis for effectiveness.

BART will include the ability to *instrument* a solve to gather enough data to draw useful conclusions about the effectiveness of acceleration schemes.

- Storage of solve parameters (eigenvalues, fluxes).
- Storage of hierarchy of iterations.
- Calculation and storage of error or residual.
- Analysis of Fourier error modes coefficients.

Adding new instrumentation must be easy!

# Outline

# The BART Code

- Deterministic, finite-element-based transport code.

# The BART Code

- Deterministic, finite-element-based transport code.
- Coded in C++, using the C++17 standard. A python script can automate creation of input files.

# The BART Code

- Deterministic, finite-element-based transport code.
- Coded in C++, using the C++17 standard. A python script can automate creation of input files.
- Uses the `deal.II` finite-element library [4].

# The BART Code

- Deterministic, finite-element-based transport code.
- Coded in C++, using the C++17 standard. A python script can automate creation of input files.
- Uses the `deal.II` finite-element library [4].
- Parallel processor (MPI) calculations are supported using PETSc [9, 8, 2].

# The BART Code

- Deterministic, finite-element-based transport code.
- Coded in C++, using the C++17 standard. A python script can automate creation of input files.
- Uses the `deal.II` finite-element library [4].
- Parallel processor (MPI) calculations are supported using PETSc [9, 8, 2].
- Uses the GoogleTest/GoogleMock framework for testing [6], with `travis.ci` continuous integration.

# The BART Code

- Deterministic, finite-element-based transport code.
- Coded in C++, using the C++17 standard. A python script can automate creation of input files.
- Uses the `deal.II` finite-element library [4].
- Parallel processor (MPI) calculations are supported using PETSc [9, 8, 2].
- Uses the GoogleTest/GoogleMock framework for testing [6], with `travis.ci` continuous integration.
- Uses the Google Protocol Buffers file format for cross-sections.

# Current support and future work

- Supports 1-, 2-, and 3-D solves.

# Current support and future work

- Supports 1-, 2-, and 3-D solves.
- Two 2nd-order formulations of the transport equation implemented: diffusion and self-adjoint angular flux equation.

# Current support and future work

- Supports 1-, 2-, and 3-D solves.
- Two 2nd-order formulations of the transport equation implemented: diffusion and self-adjoint angular flux equation.
- Acceleration methods in development: nonlinear diffusion acceleration (NDA).

# Current support and future work

- Supports 1-, 2-, and 3-D solves.
- Two 2nd-order formulations of the transport equation implemented: diffusion and self-adjoint angular flux equation.
- Acceleration methods in development: nonlinear diffusion acceleration (NDA).
- Future methods: transport two-grid acceleration (TTG), and a combination of NDA and TTG.

# Current support and future work

- Supports 1-, 2-, and 3-D solves.
- Two 2nd-order formulations of the transport equation implemented: diffusion and self-adjoint angular flux equation.
- Acceleration methods in development: nonlinear diffusion acceleration (NDA).
- Future methods: transport two-grid acceleration (TTG), and a combination of NDA and TTG.
- Instrumentation in development: *in-situ* stepwise Fourier Analysis.

# Conclusion

# Conclusion

- Implementing and analyzing acceleration methods has practical challenges.

# Conclusion

- Implementing and analyzing acceleration methods has practical challenges.
- We are developing a new code aimed at helping developers of new methods.

# Conclusion

- Implementing and analyzing acceleration methods has practical challenges.
- We are developing a new code aimed at helping developers of new methods.
- We hope that the code will ease the burden of coding up new methods, and help provide good, useful data for understanding if and why the methods are worthwhile to implement in production level codes.

Thank you

# References


[1] B. T. Adams and Jim E. Morel.
A Two-Grid Acceleration Scheme for the Multigroup Sn Equations with Neutron Upscattering.
*Nuclear Science and Engineering*, 115(May):253–264, 1993.

[2] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith.
Efficient management of parallelism in object oriented numerical software libraries.
In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[3] E. E. Lewis and W.F. Miller, Jr.
*Computational Methods of Neutron Transport*.
American Nuclear Society, 1993.

[4] G. Alzetta et al.
The `deal.II` library, version 9.0.
*Journal of Numerical Mathematics*, 26(4):173–183, 2018.

[5] Thomas M. Evans, Kevin T. Clarno, and Jim E. Morel.
A Transport Acceleration Scheme for Multigroup Discrete Ordinates with Upscattering.
*Nuclear Science and Engineering*, 165:292–304, 2010.

[6] Google.
Googletest – Google Testing and Mocking Framework.
`https://github.com/google/googletest`.
Accessed: 2017 December 20.

[7] H. Park, D. A. Knoll, and C. K. Newman.
Nonlinear Acceleration of Transport Criticality Problems.
*Nuclear Science and Engineering*, 172(1):52–65, 2012.

[8] et al. Satish Balay.
PETSc users manual.
Technical Report ANL-95/11 - Revision 3.12, Argonne National Laboratory, 2019.

[9] et al. Satish Balay.
PETSc Web page.
`https://www.mcs.anl.gov/petsc`, 2019.

# Outline

5 Backup Slides

# Backup Slide

Backup